



DATABASE Expert

Sometimes it's the seemingly straightforward tasks that prove the most time-consuming – fortunately, **Kay Ewbank** can help. Here she shows you how to add a task to Outlook and send automated emails



WORKBENCH UPDATED

Access Workbench, a program designed to help manage Microsoft Access databases, has been updated. New features include the ability to send messages to users and shut down the database, version tracking and a new interface.

Workbench allows you to see the users and computers that are using your database and lock out new users if you need to do maintenance. Checkbox options allow you to compact, compile, decompile and back up your database and open it in exclusive mode. It's available for download and you can try it 25 times before you buy. It costs \$90 (around £48) from www.vb123.com/orders.

ALL-IN-ONE DATA MANAGER

August Wind Shareware's Data Management ToolKit can be used to manage Access, MySQL, MSDE and Microsoft SQL Server databases from a single interface. You can use it to create, delete and edit table structures, create and edit stored procedures and views and work on data. A built-in Visual Query Builder helps you build and save frequently used queries. The shareware version offers a 45-day evaluation period, and the full product costs \$80 (around £42). Visit www.augustwind.com for details.

Q I've created an Access database to handle my company's sales and enquiries system, and I'd like to be able to automate various common tasks using Access. For example, I'd like to enable people to send an email about a particular order or enquiry from within Access, rather than having to copy the email address, switch to Outlook and create the message there. I'd also like my database to be able to set up a task or reminder in Outlook to remind sales people to follow up enquiries with a phone call. Is this possible? I've put a few macros in my database, but I don't know that much about programming.

Nigel Fawcett

A It's possible to do what you want, but it involves some fairly advanced programming in Access. Using one part of Office from another part has become easier and more reliable with the latest versions of the suite, but it's probably still best not to push things to their limit if you want a completely reliable system.

Adding a task to Outlook is the easier of the two procedures, so we'll look at that first. Start by opening the database you want to work with, and open a form in design view from where you want to create the tasks. Add a command button to this form, but press Cancel when asked to choose what type of code you'd like to add. Right-click on the button, and choose Build Event from the pop-up list you're offered. You should then choose Code Builder. The Visual Basic for Applications editor will open with an empty procedure saying something along the lines of:

```
Private Sub Command24_Click()
```

```
End Sub
```

The actual name of the Command24 element will reflect what your command button is called, which may be a default name assigned by Access. Scroll to the top of the code, and you'll find the declarations section. It should say something like:

```
Option Compare Database
```

Make sure there's also a line that reads:

```
Option Explicit
```

If this line doesn't exist, you need to add it.

To use Outlook, you must also make sure you reference the Outlook Object Library. From the Tools menu, choose References. If there is no tick by the reference to Microsoft Outlook 8.0 Object Library (or Outlook 9 or 10, depending on the version of Office you're using), add it. If there is no Microsoft Outlook x.x Object Library in your list, you need to add that too. Click the Browse button, then look for the Msoutl.xlb file (where x is your version of Office). The default location is the C:\Program Files\Microsoft Office\Office folder.

You can now enter the code to create an Outlook task. To start, we'll create a procedure that creates a task with a fixed subject line and description. Go to the place in your code that starts:

```
Private Sub Command24_Click()
```

Enter the following code between the Private and the End Sub lines:

```
Dim appOut As Outlook.Application
Dim taskOut As Outlook.TaskItem
Set appOut = CreateObject("Outlook.Application")
Set taskOut = appOut.CreateItem(olTaskItem)
With taskOut
    .Subject = "Subject goes here"
    .Body = "description goes here"
    .Save
End With
Msgbox ("Task created")
```

If you save the code, return to the form and switch from design view to data view, you can test your code by clicking the button. A message box should appear telling you that you've created a task. If you then switch to Outlook, you'll see a new task with a subject line reading 'subject goes here', and a description reading 'description goes here'.

Now we can make the procedure a little more useful. Add a couple of text boxes to your form: one

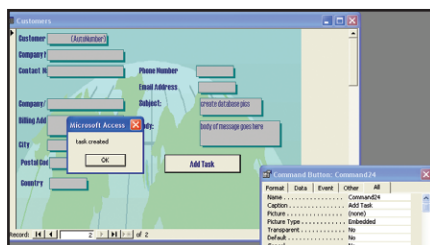
BULK EMAILING AND THE LAW

If you want to build email capabilities into a database, you need to consider the rules regarding bulk emailing. These were updated in December 2003 to address the growing spam problem. The new laws ban the sending of unsolicited marketing email unless recipients opt into a mailing list. This applies to all bulk mailing bar a few exceptional cases. Read the official guidance at www.informationcommissioner.gov.uk.

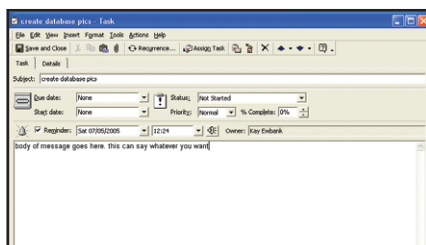
The guidance says: "You cannot transmit, nor instigate the transmission of, unsolicited marketing material by electronic mail to an individual subscriber unless the recipient of the electronic mail has previously notified you, the sender, that he consents, for the time being, to receiving such communications."

If you're wondering what the difference is between a solicited marketing message and an unsolicited message that you consent to receiving, the guide says that a solicited message is one that you have actively invited. An unsolicited message that you consent to receiving is one that you have not specifically invited but have positively indicated that you don't mind receiving. This is not the same as failing to object to receiving a message when given the opportunity to do so.

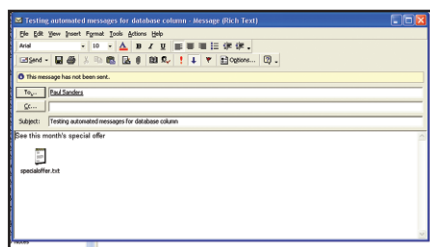
If a customer contacts you for price details, you can email them a quote. If you want to carry on emailing them, you should ask if you can send them details of other products. I'd also add a clause to each email offering to remove them from your list.



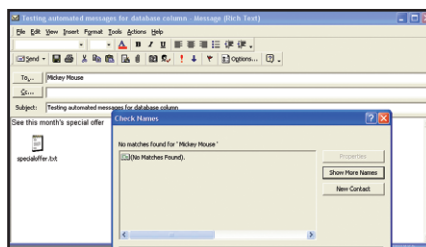
When the task is created, a message box displays in Access



The message, with the subject and body information pulled from the form, is visible in Outlook



When the email name is recognised, an email is created and placed in the Outlook inbox



If the name isn't recognised, the user gets the chance to either enter the name or type it in directly

called Subject and the other called Body. Then change the With taskout section of your code to:

```
With taskOut
    .subject = Forms!customers]
!subject
    .body = Forms!customers]!body
    .Save
End With
```

You should now be able to enter values in the Subject and Body fields on your form and have them appear in the task that is created. You can work with any of Outlook's task fields, so to set a reminder specifying the deadline for a task you could add before .Save:

```
.ReminderSet = True
.ReminderTime = DateAdd("d", 1,
Now())
```

ReminderSet = True adds a reminder and ReminderTime sets the time. DateAdd("d", 1, Now()) equals the current time plus one day. The letter m is the code for months, h for hours and n for minutes. You don't have to make your users enter the task details by hand. You could pull the values out of other fields on the form, for example. If you had a customer name field called custname and a customer phone number field called custfield, you could have something along the lines of:

```
Private Sub Command24_Click()
Dim appOut As Outlook.Application
Dim taskOut As Outlook.TaskItem
Dim subjtext As String
Dim subjbody As String
Set appOut = CreateObject("Outlook.
Application")
Set taskOut = appOut.CreateItem(olTask
Item)
With taskOut
    subjtext = "Phone " & Forms!
[customers]!Custname
    subjbody = " on phone number " & For
ms!customers]!Custphone
    .subject = subjtext
    .body = subjbody
```

```
.ReminderSet = True
.ReminderTime = DateAdd("d", 1,
Now())
.Save
End With
MsgBox ("task created")
End Sub
```

This would create a task with the subject line 'Phone John Smith' and a task description of 'on phone number 020 7999 999' or similar. You could make this more sophisticated by extracting more information from the form.

SENDING AUTOMATIC MESSAGES

Be careful with automatic email; as few things annoy people more than emails that are verging on being spam. Err on the side of caution. You may think you're passing on useful information, but imagine how your emails would appear if they were from a company you dislike. You must also follow the relevant data protection legislation. As a rule, you should add people to your mailing list only if they have actively said yes to your offer of emails or postal catalogues. See 'Bulk Emailing and the Law' opposite for details.

So how do you send an automated email to a customer? A simple way is to use a method called SendObject. This sends an email message but doesn't let you attach an external file or set aspects of the message such as its importance level.

If you want more control, you need to use a method similar to the one we employed when adding a task to Outlook. We'll assume you want to send a message with a text file attached to it, although the same technique would work for any sort of attachment. Start by creating a file called SpecialOffer.txt in a folder such as My Documents. Next, in your database, open the form from which you want to be able to send emails. Switch to design view, add a command button and choose to create a procedure attached to it using the same techniques described above. Make sure there's an Option Explicit line in the Declarations section at the top of the code, and that you've referenced the Outlook Object library.

Then enter the following code into your empty procedure:

```
Private Sub Command32_Click()
Dim objOut As Outlook.Application
Dim objOutMsg As Outlook.MailItem
Dim objOutRecip As Outlook.Recipient
Dim objOutAttach As
Outlook.Attachment
Set objOut = CreateObject("Outlook.
Application")
Set objOutMsg = objOut.CreateItem
(olMailItem)
With objOutMsg
    Set objOutRecip =
.Recipients.Add("John Smith")
    objOutRecip.Type = olTo
    .subject = "Testing auto messages"
    .body = "See this month's special
offer" & vbCrLf & vbCrLf
    .Importance = olImportanceMedium
    Set objOutAttach = .Attachments
.Add("C:\send\specialoffer.txt")
    If Not objOutRecip.Resolve Then
        objOutMsg.Display
    End If
    .Send
End With
Set objOutMsg = Nothing
Set objOut = Nothing
End Sub
```

Replace the reference to John Smith with the name of someone in your address book who doesn't mind acting as a guinea pig and getting lots of messages. You should also change the reference to C:\send\specialoffer.txt to point to your text file.

This code sets up a reference to Outlook, then creates a new mail item. Once this is created in objOutMsg, you can set any of its properties. You could add several recipients, for example; it's easy to set the subject and body of the message. Find the part of the macro that reads:

```
If Not objOutRecip.Resolve Then
    objOutMsg.Display
```

This takes care of what happens if Outlook doesn't recognise the contact name you have entered. If this is not in Outlook's address book, the message is displayed with a further message telling the user that the name hasn't been recognised, giving them the chance to choose a different name from the address book or type the email address manually.

You can use similar techniques to those we used in the previous procedure to choose names from the database form. Find:

```
Set objOutRecip =
.Recipients.Add("John Smith")
```

Instead of this, you could have:

```
custname = Forms!customers]!Custname
Set objOutRecip = .Recipients.Add
(custname) CS
```

CONTACT

KAY EW BANK

Kay is *Computer Shopper's* database expert and has worked as a consultant and author for 20 years

kay@computershopper.co.uk